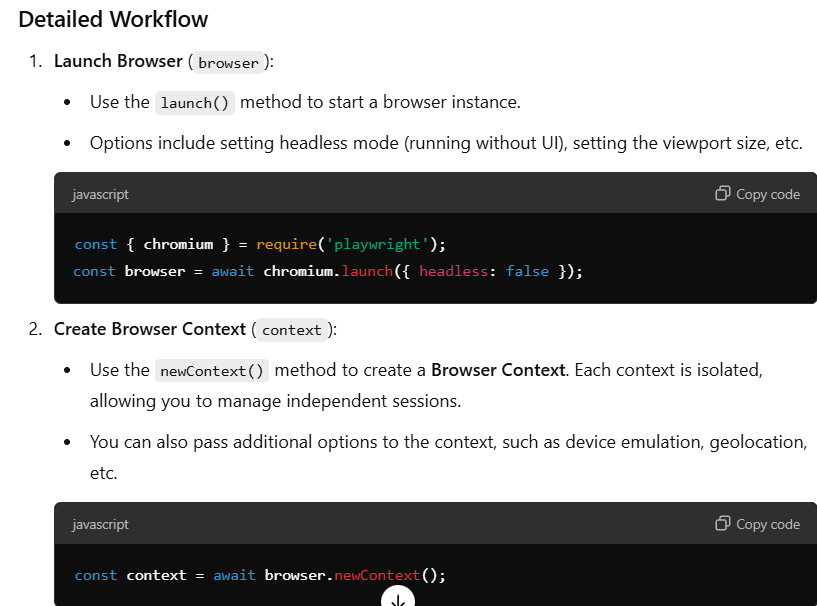
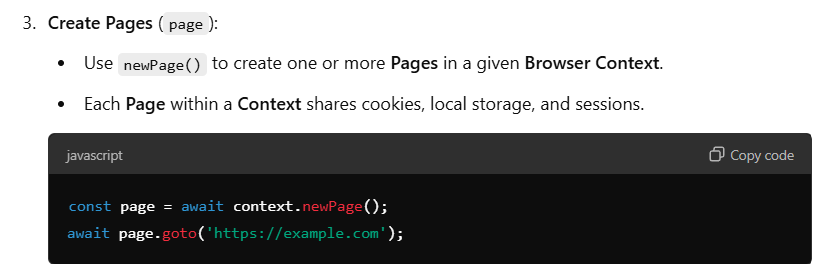
In Playwright, there is a well-defined **hierarchy** or **workflow** that you follow when working with browser automation. It starts from launching the browser and goes all the way down to performing actions on elements on the page. Let’s break down the typical **hierarchy**:

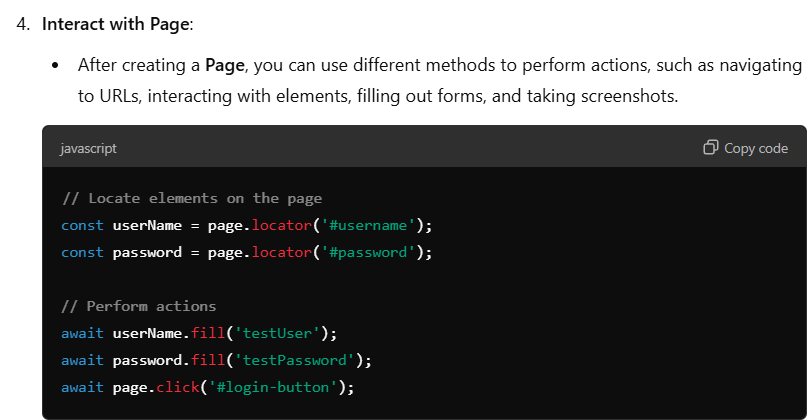
**Playwright Browser Automation Hierarchy**

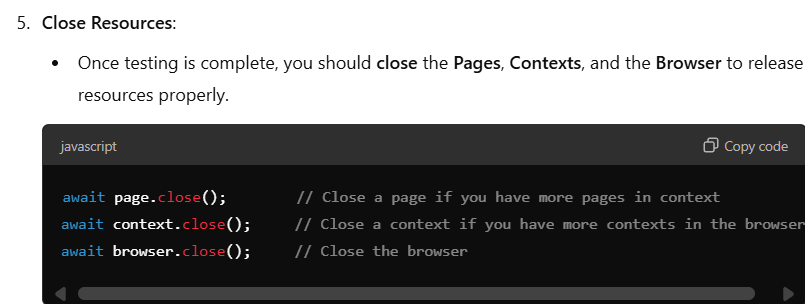
1. **Launch Browser**:
   * First, you launch a browser instance. Playwright supports different browser engines such as Chromium, Firefox, and WebKit.
2. **Create Browser Context**:
   * After launching the browser, you create a **Browser Context**. A browser context acts like an isolated environment or session within the browser, allowing for isolated browsing sessions (similar to incognito windows).
3. **Create Pages within the Context**:
   * Within a **Browser Context**, you can create one or more **Pages**. Pages are equivalent to tabs in a browser. All pages in a given context will share the same session cookies and storage.
4. **Interact with Page**:
   * After creating a **Page**, you can perform actions on the page—such as navigation, clicking buttons, filling forms, verifying content, etc.
5. **Close Resources**:
   * Close **Pages**, **Contexts**, and the **Browser** after your actions are completed to release resources.

Let’s go through each of these steps in a typical workflow:









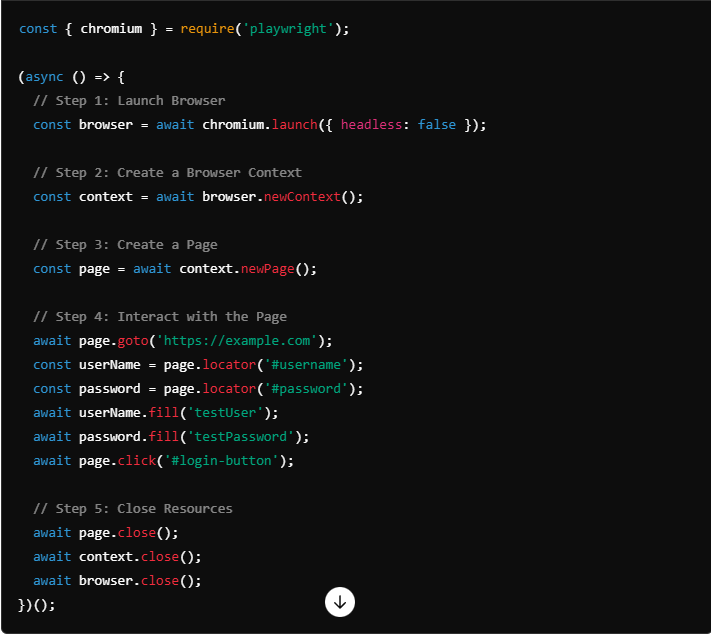
**Hierarchy Representation**

Here’s how the hierarchy looks, step by step:

1. **Browser** (browser)
   * browser = await chromium.launch();
     + Top-level entity representing the browser.
2. **Browser Context** (context)
   * context = await browser.newContext();
     + Represents an isolated environment/session.
     + You can create multiple **Contexts** for different users.
3. **Page** (page)
   * page = await context.newPage();
     + Represents a tab within a **Browser Context**.
     + You can create multiple **Pages** in the same **Context** to represent multiple tabs for the same user.
4. **Interactions on Page**
   * Actions such as **goto(), click(), fill(), type(), screenshot(), etc.** to automate browser actions.

**Example Putting It All Together:**

Here’s a complete example:



**Summary of Hierarchy:**

1. **Launch Browser** (browser):
   * Start a browser instance using chromium.launch().
2. **Create Browser Context** (context):
   * Use browser.newContext() to create a separate session.
3. **Create Pages** (page):
   * Use context.newPage() to create a page/tab within that context.
4. **Perform Interactions** on the **Page**:
   * Use various methods like .goto(), .click(), .fill() to interact with the webpage.
5. **Close** Pages, Contexts, and Browser:
   * Close them in the order of **Page**, **Context**, and finally **Browser** to release resources properly.

By following this hierarchy, you can efficiently manage browser sessions and interactions, ensuring test isolation and avoiding interference between different scenarios.